DOCUMENT RESUME

ED 224 691 SE 039 730

AUTHOR	Moser, James M.; Carpenter, Thomas P.
TITLE	Using the Microcomputer to Teach Problem-Solving Skills: Program Development and Initial Pilot Study. Report from the Project on Studies in Mathematics. Working Paper No. 328.
INSTITUTION	Wisconsin Center for Education Research, Madison.
SPONS AGENCY	National Inst. of Education (ED), Washington, DC.
PUB DATE	Jul 82
NOTE	73p.
PUB TYPE	Reports - Research/Technical (143)
EDRS PRICE	MF01/PC03 Plus Postage.
DESCRIPTORS	Addition; Computer Programs; Computers; Educational Research; Educational Technology; *Elementary School Mathematics; *Grade 1; Mathematics Instruction; *Microcomputers; Primary Education; *Problem Solving; Subtraction
IDENTIFIERS	*Mathematics Education Research; *Word Problems

ABSTRACT

This document reports on the initial phase of a project investigating how to relate formal mathematical representational and problem solving skills to informal strategies that children naturally invent to solve simple addition and subtraction problems. A program was developed that allows pupils to solve word problems on a microcomputer. A pilot study was carried out with four first-grade children. The subjects were individually instructed for a series of nine 20-minute lessons. The results of the study indicated that the program is effective in teaching representational and problem-solving skills. Before instruction, the subjects consistently wrote incorrect sentences for incorrect problems and generally did not use their number sentences for their solutions. Following instruction, three of the four children consistently used number sentences to solve a wide variety of addition and subtraction problems. It is concluded that further investigation seems warranted, and that this pilot investigation suggests that microcomputers can have important roles in instruction. (MP)

WORKING PAPER NO. 328

USING THE MICROCOMPUTER TO TEACH PROBLEM-SOLVING SKILLS: PROGRAM DEVELOPMENT AND INITIAL PILOT STUDY

by James M. Moser and Thomas P. Carpenter

Report from the Project on Studies in Mathematics

Thomas P. Carpenter
Faculty Associate

James M. Moser
Senior Scientist

Wisconsin Center for Education Research
The University of Wisconsin
Madison, Wisconsin

July 1982

The research reported in this paper was funded by the Wisconsin Center for Education Research which is supported in part by a grant from the National Institute of Education (Grant No. NIE-G-81-0009). The opinions expressed in this paper do not necessarily reflect the position, policy, or endorsement of the National Institute of Education.

Wisconsin Center for Education Research

MISSION STATEMENT

The mission of the Wisconsin Center for Education Research is to understand, and to help educators deal with, diversity among students. The Center pursues its mission by conducting and synthesizing research, developing strategies and materials, and disseminating knowledge bearing upon the education of individuals and diverse groups of students in elementary and secondary schools. Specifically, the Center investigates

- diversity as a basic fact of human nature, through studies of learning and development
- diversity as a central challenge for educational techniques, through studies of classroom processes
- diversity as a key issue in relations between individuals and institutions, through studies of school processes
- diversity as a fundamental question in American social thought, through studies of social policy related to education

The Wisconsin Center for Education Research is a noninstructional department of the University of Wisconsin-Madison School of Education. The Center is supported primarily with funds from the National Institute of Education.

Table of Contents

Table of Contents (continued)

List of Tables

List of Figures

Abstract

This report documents the initial phase of a project investigating how to relate formal mathematical representational and problem-solving skills to the informal strategies that children naturally invent to solve simple addition and subtraction word problems.

The microcomputer provides a means for directly relating formal symbolic representations to children's informal modeling processes. A program has been developed that allows children to use a micro-computer rather than physical objects to solve word problems. Children initially are taught to use the microcomputer to solve simple word problems using essentially the same processes that they use with physical objects. They produce sets of objects one at a time and can make a single set, or make two sets, or remove elements from a set they have constructed. The objectives of these initial activities are to familiarize children with the microcomputer and make the transition from using physical objects to using the pictorial display.

The connection between the informal modeling processes and the formal mathematical symbolic representations is made by teaching the children that they do not have to construct sets on the microcomputer one element at a time; they can construct them by writing number sentences. To solve an addition problem, they enter an addition sentence like $8 + 5 = \square$. This actually produces a set of 8 and a set of 5, just as the child would

using physical objects. Entering a subtraction sentence $13 - 8 = \square$ produces a set of 13 and then removes 8 elements to another portion of the screen. Since the number sentence that the children enter actually constructs the physical representation that they can use to solve the problem, writing the number sentence becomes part of the solution process, not an unrelated activity.

A pilot study was carried out with four first-grade children. The children were individually instructed for a series of nine 20-minute lessons. The results of the pilot study indicate that the program is effective in teaching representational and problem-solving skills. Before instruction, the four children consistently wrote incorrect sentences for more difficult problems and generally did not use their number sentences for their solutions. Following instruction, three of the four children consistently used number sentences to solve a wide variety of addition and subtraction problems.

Introduction

The purpose of this report is to describe the results of the initial phase of the microcomputer research project carried out by the Mathematics Work Group of the Wisconsin Center for Education Research. The aim of the project is to investigate the transition phase in children's learning of symbolic representational skills in mathematics as they progress from the informal strategies learned independent of school instruction to the more formal skills of writing symbolic sentences to represent verbal problems and then solving those sentences. Addition and subtraction problems are the content domain of the project. The project uses the microcomputer to establish a direct link between writing symbolic number sentences and children's informal modeling processes.

This report covers work carried out during the period January 1982 to June 1982. A program was developed for the Apple II microcomputer and then used in a teaching experiment with four first-grade children from a private school in Madison, Wisconsin. Subsequent sections contain the background and rationale for the study, a description of the computer program, the instructional treatment used in the teaching experiment, and the results of the study. A final section presents some overall conclusions together with projections for future directions of the research project.

Background

In the last few years a substantial body of research focused on the learning of addition and subtraction concepts in general and on the solution of addition and subtraction word problems in particular. In the fall of 1979, an international conference devoted exclusively to the study of addition and subtraction was held (Carpenter, Moser, & Romberg, 1982), and several major reviews of work in this area have been written (Carpenter, Blume, Hiebert, Anick, & Pimm, in press; Carpenter & Moser, in press; Riley, Greeno, & Heller, in press).

Current research on children's solution of basic addition and subtraction word problems follows a basic cognitive approach outlined by Glaser and Pellegrino (1978). This approach involves the detailed analysis of a specific content domain which is then related to a careful analysis of the strategies that children use to solve problems within the domain. Currently, there is good agreement regarding the basic characterization of addition and subtraction word problems, and there is a reasonably consistent picture of the difficulty level of different types of problems and the informal problem-solving strategies children invent independently of instruction. However, relatively little is known regarding the transition from these informal strategies to the formal addition and subtraction skills taught in school (Riley et al., in press).

Analysis of Problem Types

Early research took several approaches to the characterization of word problems. One was to classify problems in terms of syntax, vocabularly level, number of words in a problem, and so forth (e.g., Jerman, 1973; Suppes, Loftus, & Jerman, 1969). A second approach differentiated between problems in terms of the open sentences they represented (e.g., Grouws, 1972; Rosenthal & Resnick, 1974). The most productive approach, that followed by current research, is based on the semantic characteristics of problems (Carpenter & Moser, 1982; Gibb, 1956; Greeno, 1978; Vergnaud, 1982). Semantic analysis is based primarily on structural characteristics involving the action or relationship described in the problem. Altogether there are six basic semantic problem types: Separate, Combine, Compare, Join, and two types of Equalize problems (Carpenter & Moser, 1982). The following four subtraction problems illustrate the kinds of distinctions drawn between problem types. Although all four problems can be represented by the mathematical sentence 12 - 5 = $\square$, they represent distinct interpretations of subtraction:

> Tim has 12 candies. He gave 5 candies to his sister. How many candies does Tim have left?
>
> Tim has 12 candies. Five the them are grape and the rest are lemon. How many lemon candies does Tim have?
>
> Tim has 5 candies. His sister Connie had 12 candies. How many more candies does Connie have than Tim?
>
> Tim has 5 candies. His sister Connie gave him some more candies. Tim has 12 candies. How many candies did Connie give to him?

The first problem, Separate, describes the action of removing a subset of a given set. The second, Combine, is a static situation in which one of two parts of a known whole must be found. The third problem, Compare, involves the comparison of two distinct sets. The fourth, Join, describing an additive change action, has as its unknown the size of that change. For each semantic problem type, three distinct problems can be generated by varying which quantity is unknown. The first problem above could be altered as follows to produce a parallel missing minued problem:

> Tim had some candies. He gave 5 candies to his sister. If he has 7 candies left, how many candies did he have to start with?

As can be seen from these examples, a number of semantically distinct problems can be generated by varying the structure of the problem, even though many of the same words appear in the different versions.

Analysis of Children's Performance

Most past studies of addition and subtraction were limited to finding out which types of problems were most difficult. More recently, work has begun to focus on the processes children use to solve different problems. Measuring response latencies (Groen & Parkman, 1972; Groen & Resnick, 1977; Suppes & Groen, 1967; Woods, Resnick, & Groen, 1975) or conducting clinical interviews (Blume, 1981; Brush, 1978; Carpenter, Hiebert, & Moser, 1981; Carpenter & Moser, 1982; Hiebert, 1981; Lindvall & Ibarra, 1980), researchers have identified a number of strategies that children use to solve different addition and subtraction problems.

Data from these studies suggest that, contrary to popular notions, young children are relatively successful at analyzing and solving simple verbal problems. Before receiving formal instruction in addition and subtraction, young children invent informal modeling and counting strategies for solving addition and subtraction problems (Carpenter, Hiebert, & Moser, 1981; Carpenter & Moser, 1982). These results suggest that word problems may be the most appropriate context for introducing formal concepts of addition and subtraction. The present research investigates this hypothesis.

The informal solution strategies that children invent have a clear relationship to the addition and subtraction problem types described above. At the earliest stage most children directly model quantities described in a problem, perform actions on these models, and enumerate sets to determine an answer. For example, to solve the following Join, missing addend problem, children at this stage generally would construct a set of 5 objects, add more objects until there was a total of 12 objects, and count the number of objects added.

> Sally has 5 baseball cards. How many more baseball cards does she need to have 12 baseball cards altogether?

At the next stage, children shift to more abstract counting strategies. To solve the above problem, a child would recognize that it was unnecessary to construct the set of 5 objects and instead simply count from 5 to 12, keeping track of the number of counts. At both stages, the type of strategy used depends upon the semantic structure of the problem, suggesting that children do not transform problems to a single representation of addition or subtraction. During the early stages of development, children do not appear to recognize the interchangeability of their strategies. In other words, they do not initially recognize that either a separating or an adding on strategy will generate the same solution. A completely developed concept of addition and subtraction presumably would require an integration of various interpretations of those operations as represented by the different counting strategies. That is to say, the concrete counting strategies should eventually evolve into the abstract representation of formal mathematics (Carpenter & Moser, 1982).

Current instruction clearly fails to build upon the informal strategies that children develop outside of school. There is reasonable consensus on how children solve addition and subtraction problems, but there is a great gap between what is known regarding children's solution processes and current instructional practice (Carpenter, 1981).

Representing Addition and Subtraction Problems

The process of representing a real world or verbally posed problem is a fundamental problem-solving skills. The development of this skill is a major objective of the entire mathematics curriculum. We hypothesize that a cause of the difficulty of older children to solve problems (Carpenter et al., 1980) may be their inability to adequately represent a given problem with the appropriate mathematical symbolism. The contrast between young children's success in analyzing simple problems and older children's performance on more complex problems suggests that the transition from simple representations such as physical modeling, counting, and tallying to symbolic mathematical representations and operations such as writing number sentences, memorizing facts, and using algorithmic procedures is a critical stage in children's learning of mathematics in general, and of problem-solving skills in particular (Carpenter, 1981).

A key aspect of the transition from solving problems using informal procedures based upon simple representational skills to a formal mathematics approach is writing mathematical symbols to represent the problem and its components. The skill of symbolic representation is one of the major objectives of elementary school instruction. Writing mathematical expressions to represent a problem situation is a skill fundamental to problem solving from elementary arithmetic to advanced mathematics. The association of real world problems with abstract mathematical representations takes place in many areas of mathematics; addition and subtraction, multiplication, rational number, geometric congruence, and similarity are several examples that can be cited.

At the time children are first introduced to writing mathematical sentences to help solve word problems, their informal strategies and procedures make more sense to them. As a consequence, they see no connection between the two activities, although most children eventually learn to write number sentences to represent simple problems and are able to solve the problems using their informal modeling and counting strategies. The operations represented by the number sentences are often inconsistent with the modeling and counting strategies used to solve the problem. Writing a number sentence is something that young children do for the teacher, something they often perceive as unrelated to the solution of the problem. This is not surprising. The children already know how to model the problem physically. Until they have memorized the basic facts and learned computational algorithms, writing a number sentence does not help them solve the problem.

In a study investigating the effects of initial instruction on the processes children used to solve basic addition and subtraction verbal problems, Carpenter, Moser, and Hiebert (1981) considered the role of writing number sentences in the solution process. Prior to instruction

43 first-grade children were individually tested on a variety of addition and subtraction word problems. After a two-month introductory unit on addition and subtraction, the children were retested. On the posttest most children could write number sentences to represent addition and subtraction problems. However, very few recognized that the arithmetic sentence was a mechanism that they might use to help them solve the problem. Once they had written a sentence, most children appeared to ignore it and used the semantic structure to decide on a solution strategy. In fact, in spite of instructions to the contrary, about a fourth of the subjects solved a problem before writing a sentence. When children wrote an incorrect sentence but computed the correct answer, they would often complete the open sentence with their answer. The fact that sentence writing did not influence children's solution processes suggests a lack of coordination by the children between the two processes.

Description of the Computer Program

Overview

We have developed a program that allows children to use a microcomputer rather than physical objects to solve word problems. Children initially are taught to use the microcomputer to solve simple word problems using essentially the same processes that they use with physical objects. They produce sets of objects one at a time by pushing the $\boxed{\rightarrow}$ key. They can make one set, or make two sets, or remove elements from a set they have constructed. The objectives of these initial activities are to familiarize children with the microcomputer and make the transition from using physical objects to using the video display.

• The connection between the informal modeling processes and the formal mathematical symbolic representations is made by teaching the children that they do not have to construct sets on the microcomputer one element at a time; they can construct them by writing number sentences. To solve an addition problem, they enter an addition sentence like $8 + 5 = \square$. As well as the number sentence, this actually produces a set of 8 and a set of 5, just as the child would using physical objects or the arrow key on the computer. Entering a subtraction sentence $13 - 8 = \square$ produces a set of 13 and then removes 8 elements to another portion of the screen. Students also learn to write open addition sentences to represent and to solve certain kinds of word problems. Since the number sentence that the children enter actually actually constructs the pictorial representation that they can use to solve the problem, writing the number sentence becomes part of the solution process not an unrelated activity. Figure 1 illustrates the video display resulting from entry of the sentence $7 + 28 = \square$.

Details of the Program

The major feature of the computer program is the ability to enter onto the video display pictorial and symbolic configurations by depressing appropriate keys on the keyboard. The video display is arranged in three adjacent sectors which can be thought of as corresponding to the elements of the number sentence $a + b = c$ or $a - b = c$. Entry of configurations

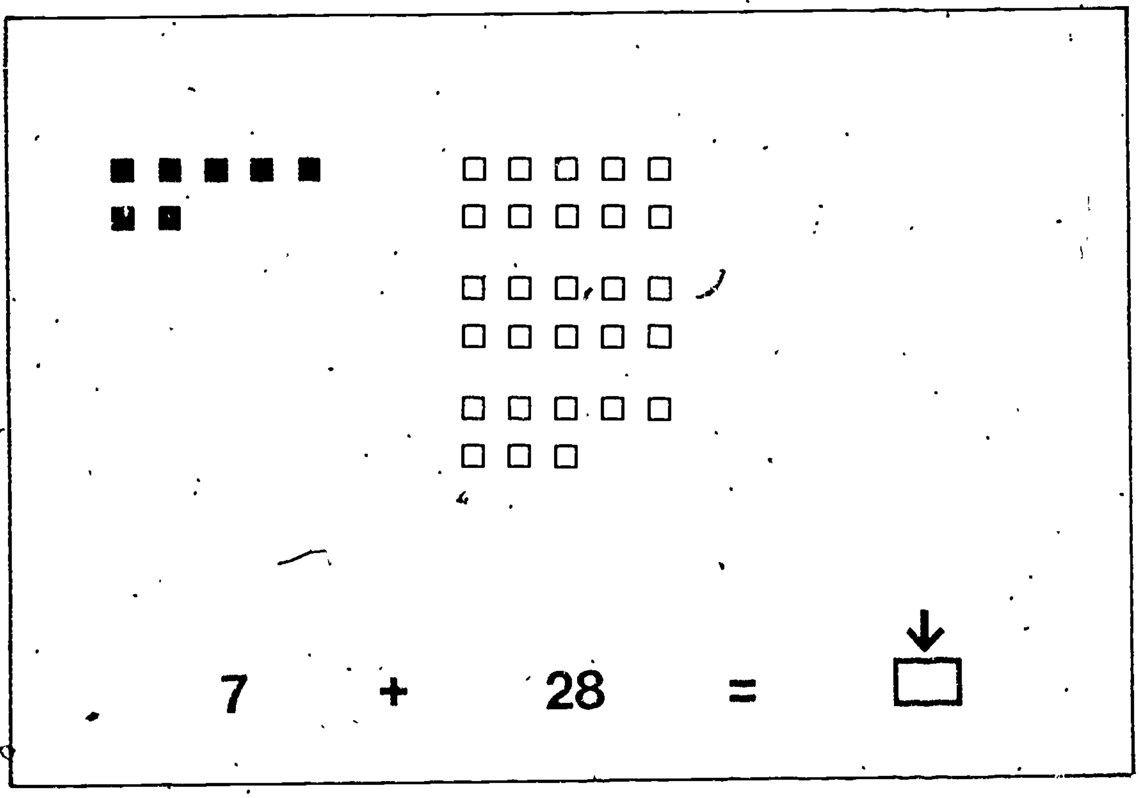

Figure 1. Video screen display after entry of 7 + 28 = □.

into a particular sector is denoted by a small arrow, with the head pointing upward for sectors a and b, and the head pointing downward for sector c. Pictorial configurations can be entered only in sectors a and b, and they appear in the upper two-thirds of the sector. The bottom one-third is reserved for symbolic entries. On initial use of the program, or after clearing the video display, the indicator arrow for entry of pictorial or symbolic configurations automatically returns to the a sector. Movement to the b or c sector is carried out by several means described later. Once movement to the right on the display is made, that is from a to b or from b to c, movement to the left is impossible. One has to begin anew in sector a, either by rebooting the entire program or more simply by depressing the ESC(ape) key. In either case, the entire video display is cleared of all configurations.

The computer program used in the study was written in Apple Pascal using the Pascal ANIMATION Package from Apple Special Delivery software to help create a large character set and handle some of the display tasks. It requires a 48K Apple II+ with an extra 16K RAM card in slot Ø for operation, and a color monitor.

The program consists of two texts files, BOXES1.TEXT and BOXES2.TEXT. Two special libraries, ANIMATION and CRTSTUFF, are included in the system library. ANIMATION comes with the Pascal ANIMATION Package and CRTSTUFF is a special library of CRT handling routines. The system disk also contains the large character font, BOXES.FONT, which is four times the size of normal Apple characters. A listing of the program is contained in Appendix A.

Pictorial configuration. The pictorial configurations consisted of small squares arranged in a pattern resembling the TILE configurations used in Japanese elementary mathematics education (Hatano, 1982). Squares appear in horizontal rows of at most five elements with twice as much vertical spacing between the second and third and between the fourth and fifth rows as between the first and second, third and fourth, and fifth and sixth rows. This visual emphasis of groups of ten was designed to make counting the squares easier for children who recognized the configural patterns. Space limitations on the video display and the desire to make the squares large enough that children could visually discriminate among the squares allowed a maximum of 30 squares for each of sectors *a* and *b*. Squares in sector *a* were blue, while those of sector *b* were green.

Entry of pictorial configurations is made in two different ways. The first method provides for a one-by-one incremental entry of squares by means of successive depressions of the [→] key. Accompanying this entry of squares is the display of the corresponding number in the lowest third of the sector. One has the option of omitting the concurrent display of the numeral. Removal of one or more squares from a configuration in a sector is carried out by depression of the [←] key. Before entry, the symbolic display is empty, and the numbers 1, 2, 3, etc. appear as the child depresses the [→] key. Once this has been done and the child elects to remove the entered squares by depressing the [←] key, the numerals go down in order. If all are removed the numeral "0" is displayed. As mentioned above, the maximum number of squares that can be entered in a sector is 30. If the child attempts to enter more than 30 by continued depression of the $\boxed{\rightarrow}$ key, a "beep" is heard. Similarly, attempted removal of more squares than are present causes the "beep" once zero has been reached. Initially the arrow is in sector *a* and squares are added to that sector. To add squares to sector *b*, the space bar is depressed. This causes the arrow to move to sector *b* and any subsequent operations on the $\boxed{\rightarrow}$ or $\boxed{\leftarrow}$ keys result in squares being added to or removed from sector *b*.

The second method of producing pictorial configurations is by the depression of numeral keys in the upper row of the computer keyboard. If a two-digit number, such as 14, is entered by successive depressions of the "1" and "4" keys, then the corresponding number of squares is automatically produced in the desired sector, the squares appearing rapidly in one-by-one succession. If a one-digit number is chosen, then the visual display is not produced until the child "informs" the computer that the digit depressed is the number of squares desired and not the ten's digit of a two-digit number. Means available for transmitting this information are depression of any of the following: space bar, +, -, RETURN, or in the instance where the configuration is desired in the *b* sector, the = sign. A configuration entered with numeral keys can be subsequently incremented and decremented by depressing the $\boxed{\rightarrow}$ and $\boxed{\leftarrow}$ keys, respectively.

The representation of a take-away action is brought into play by initially producing a configuration in sector *a* by either of the two means described above. It is activated by the depression of the – key which produces a small arrow pointing in the right hand direction. The arrow is located in the lower one-third of the video display midway between sector *a* and sector *b*. If simultaneous display of numerals with the configuration of squares is called for, then the right-pointing arrow also has a minus sign below it; if no numeral production is called for, then the minus sign does not appear. In either case, this feature causes squares from sector *a* to be moved to sector *b*. Once the – key has been depressed, the movement can be effected on a one-by-one basis by successive depressions of the $\boxed{\rightarrow}$ key, or on an automatic, rapid one-by-one basis by depression of numeral keys. The same procedures for two- and one-digit numbers described above operate here. Corrections or adjustments to the size of the configuration in sector *b* can be made by depression of either the $\boxed{\rightarrow}$ key, in which case another square will be moved from sector *a* to sector *b*, or the $\boxed{\leftarrow}$ key, in which a square will be returned back to sector *a* from sector *b*.

When the concurrent display of numerals below the configurations of squares is called for, the numeral below the configuration of sector *a* remains constant, that being the number of squares in the initial configuration. The numeral below the configuration of squares in sector *b* varies, depending on how many squares are present there. Thus, the appropriate subtraction number sentence will be displayed. If some set of squares has been removed from the sector *a* to sector *b* and then later returned by means of the $\boxed{\leftarrow}$ key, the numeral 0 will appear in sector *b* when all squares have been returned. Attempts to move more squares from one sector to another, in either direction, than is possible to move, will result in production of the "beep." For example, if the original configuration in sector a had 12 squares and the child then depressed "- 15," the computer would react with a "beep."

Symbolic configurations. The standard mathematical symbols for numbers, operations (+ or -), equality (=), and an unknown quantity (□) can be produced only if the initial menu selection calls for such production. These symbols appear in white in the lower one-third of the video display. The program calls for the production of only numerals in sectors a, b, and/or c, or of a complete number sentence that is essentially correct in form. Incomplete sentences such as 5 + 7 □ or 5   7 = □ would not appear. Sentences that are impossible to solve within the domain of whole numbers such as 13 - 15 = □ or 9 + □ = 3 would also not be accepted by the computer.

Numerals can be generated immediately upon depression of appropriate numeral keys on the keyboard, with numbers in sectors a and b being restricted to 30 or less. Production of a numeral up to 60 in sector c can be carried out only by depression of numeral keys on the keyboard. Pictorial configurations are not produced in sector c.

Mathematical sentences of the form $a + b = c$ or $a - b = c$ may be entered in the computer. The sentences may be closed, in that a number stands in the place of a, b, and c, or open in that a □ to represent a missing number may stand in the place of a, b, or c. An a priori decision was made to not allow the noncanonical subtraction sentences □ $- b = c$ or $a -$ □ $= c$.

The missing number box (☐) is entered by the depression of SHIFT. (The entry of = and + also require prior depression of the SHIFT key.) In practice, the child "writes" a complete mathematical sentence by choosing a numeral for position *a*, then depressing the + or − key, which causes the pictorial configuration in sector *a* to appear if it is not already there by reason of representing a two-digit number. It also causes the upward-pointing indicator arrow to move to sector *b*. At that time the + or − symbol is also displayed on the screen. If the − symbol is chosen, the right-pointing arrow is also shown above the symbol. Next, the numeral for the *b* position is depressed followed by =, which causes the pictorial configuration in sector *b* to appear subject to conditions described in earlier paragraphs. This also produces the appearance of the = symbol in its proper position in the sentence as well as the movement of the downward pointing arrow to sector *c*. At this point the child may either enter a third numeral, or, if not entered in another position in the sentence, the ☐ to represent an unknown. The computer will accept computationally incorrect sentences such as 5 + 3 = 9 without interacting with the student. The ☐ may be entered only in the *c* position for subtraction sentences, and in any one of the three positions—*a*, *b*, or *c*—for addition sentences. If the ☐ is entered in any position and the indicator arrow has not been moved to a different sector, the ☐ may be overridden by entry of a numeral. If the ☐ is entered in the *a* or *b* position and the indicator arrow is moved to another sector, no display of a pictorial configuration is produced above the ☐. Only one ☐ per sentence may be entered.

Provision is made for the child to correct an incorrect entry at any time in the production of a sentence. By depressing the X key, the most recent entry is removed and that portion of the video display is cleared. In the case where the error is made in the entry of a numeral in the *b* position after the - key has been depressed, all the squares that had been moved from sector *a* to sector *b* are returned to sector *a*, with the - symbol remaining.

The Teaching Experiment

Following development of the computer program, a teaching experiment was carried out. The experiment was carried out in order to (a) validate the physical and conceptual features of the microcomputer program, (b) study in some detail the development of sentence writing ability, (c) develop and validate procedures of instruction related to use of the specific computer program developed, and (d) evaluate the effectiveness of the computer program and the related instructional procedures in linking the informal solution strategies of young children and the formal symbolism of mathematics.

Subjects for the teaching experiment were selected from a group of first-grade children by applying selection criteria described below. A series of lessons were taught to four selected subjects on an individual basis by one of the two experimenters in the presence of a second adult observer. Following instruction, a brief individually administered problem-solving interview was given to each subject. The experiment and the posttesting were carried out during a period of approximately five weeks in April and May 1982.

Subjects. The subjects for the study were selected from the two first-grade classes in a parochial school serving a middle-class neighborhood in Madison, Wisconsin. The teachers recommended 11 children who were in the middle range of ability for their classes and were reasonably good at explaining their ideas. Individual interviews were conducted with these 11 children requiring them to solve a variety of addition and subtraction verbal problems and to perform some counting tasks. The set of screening tasks is given in Table 1. For the first nine verbal problems, a set of plastic cubes was available for the child to use. For problems 7, 8, and 9, paper and pencil were also provided, with the direction to write a number sentence prior to solving. Problems 10 and 11 were designed to assess whether children could use counting-on procedures. Cubes were not provided for these problems.

The individual interviews were conducted in mid-April. Attention was given to a child's ability to express him/herself and give clear explanations of procedures used to solve problems as well as to the actual processes used to solve the given problem. Selected subjects did not use memorized number facts, generally employed direct modeling procedures, demonstrated the ability to count forward from a beginning number other than "one," and were unable to write appropriate number sentences for problems 8 and 9. As a result of the screening, two white male subjects, Jack and Roger, and two white female subjects, Helen and Kathy, were chosen to participate in the teaching experiment. Their success on the

Table 1

Subject Screening Tasks

Verbal Problems	
1. Join Norman had 6 books. His friend gave him 9 more books. How many books did Norman have altogether?	6. Join, missing addend Robert has 8 pet fish in his tank. How many more fish does he have to put in the tank so there will be 14 fish altogether?
2. Separate Jeanne has 13 buttons. She gave 9 buttons to Evelyn. How many buttons did Jeanne have left?	7. Separate There were 11 strawberries growing on a bush. Alex picked 8 of them. How many berries were left growing on the bush?
3. Compare Robert has 3 marbles. Dorothy has 8 more marbles than Robert. How many marbles does Dorothy have?	8. Join, missing addend Kathy has 9 stamps. How many more stamps does she have to put with them to have 15 stamps altogether?
4. Separate, missing minuend There were some birds sitting on a wire. Four of the birds flew away. Then there were 7 birds left. How many birds were there sitting on the wire before any flew away?	9. Compare Joe won 9 prizes at the fair. His sister Connie won 13 prizes at the fair. How many more prizes did Connie win than Joe?
5. Compare Ellen has 7 halloween candies. Her friend Greg has 12 halloween candies. How many more candies does Greg have than Ellen?	10. Join, missing addend Ralph has 7 marbles. How many more marbles does he have to put with them to have 11 marbles altogether?

Counting Tasks

11. a) Can you count forward, starting at 15 and ending at 20?

    [Say: "I am going to start counting at 8 and count up 5 more numbers: 8, 9, 10, 11, 12, 13."]

    b) Can you start at 6 and count on three more numbers?

    c) Can you start at 9 and count up six more numbers?

screening tasks is shown in Table 2, where favorable results on the initial nine verbal-problem tasks reflect their choice of an appropriate solution strategy, even if execution of that strategy may have included a counting or computational error.

Lessons. A series of nine 20- to 30-minute individual lessons were developed. Since the primary concern of the project was the study of symbolic representations of verbal problems, each lesson was designed to have as one of its components the opportunity to solve a variety of verbal problems.

A great deal of flexibility was employed during the lessons, depending upon individual differences and day-to-day variations in a child's ability to attend to the learning tasks. Thus, for an individual child, the aims of a particular lesson may have been completed either earlier or later than planned.

Lesson 1. The lesson started with a general introduction to the computer, including how to turn it on, load the program, and select from the menu. For this lesson, no symbols were shown in the bottom one-third of the video display. Children learned the function of the $\boxed{\rightarrow}$, $\boxed{\leftarrow}$, –, ESC(ape) keys, and the space bar. children were asked to count the number of squares displayed in either sector *a* or *b* of the screen or in both. Some simple verbal problems were presented and the child was asked to use the computer and its display of squares to help solve the problems.

Lesson 2. Again, no symbols were displayed. The general aim was a review of material from the previous lesson. Numbers in the high teens

Table 2

Results on Pre-Instruction Screening Tasks

Task		Jack	Roger	Helen	Kathy	Total
		Subjects				
1.	Join	+	+	+	+	4
2.	Separate	+	+	+	+	4
3.	Compare	+	+	-	-	2
4.	Separate, missing minuend	+	+	+	+	4
5.	Compare	+	+	+	+	4
6.	Join, missing addend	+	+	+	+	4
7.	Separate: solve	+	+	+	+	4
	write sentence	+	+	+	+	4
8.	Join, missing addend: solve	+	+	+	+	4
	write sentence	-	-	-	-	0
9.	Compare: solve	+	+	+	+	4
	write sentence	-	-	-	-	0
10.	Join, missing addend	+	+	NA[a]	+	3
11a.	Count forward, 15 to 20	+	+	NA	+	3
11b.	Count on 3 from 6	+	+	NA	+	3
11c.	Count on 6 from 9	+	-	NA	+	2

Note: Wording of tasks is given in Table 1. + indicates the use of an appropriate procedure; -, an inappropriate procedure.

[a]Not administered.

and twenties) were introduced and a greater variety of word problem types were used, with an emphasis on the Join, missing addend problem.

Lesson 3. The major point of this lesson was the introduction of the numerical symbols in the video display, which were used for all remaining lessons. Initially, sets of squares (and the accompanying numerals) were generated and manipulated by $\boxed{\rightarrow}$, $\boxed{\leftarrow}$, and - keys and the space bar. Later in the lesson, the generation of sets of squares by simply depressing the numeral keys in the upper row of the computer keyboard was presented.

Lesson 4. Much of the lesson was devoted to review of the work of the previous lesson. The latter portion of the lesson included the introduction of the + and = symbols. Both required showing the child how to use the SHIFT key first.

Lesson 5. Writing a complete, closed sentence was the objective of this lesson. As in all previous lessons, a sampling of the variety of types of verbal problems was included. There was nothing very new in this lesson, although the child received suggestions to seek efficient ways of counting the displays of squares representing the solution.

Lesson 6. This lesson aimed to continue the practice of skills learned in earlier lessons. Numbers in the late teens and twenties were used for a variety of word problems.

Lesson 7. In this lesson, writing of open sentences was taught as the child learned how to enter the $\square$ in the sentence. For the most part, problems resulting in canonical addition and subtraction sentences were used. Some noncanonical situations were used.

Lesson 8. This was essentially a review and consolidation of previously learned skills, with the emphasis on using the $\square$ in the open sentence to represent the unknown. A variety of problem situations were used including a number of non-canonical ones.

Lesson 9. This final lesson was again a review and consolidation of previous lessons.

The lessons were conducted on an individual basis by one of the two principal investigators. With the exception of one lesson with one child, all lessons were observed by a second person who acted as recorder of the lesson. The average lesson took approximately 20 minutes. For the earlier lessons, several days intervened between lessons whereas during the latter portion of the experiment, lessons occurred on an almost daily basis. Lesson dates are given in Table 3 for all four children.

Problem-Solving Interviews

On the day following the last lesson, a follow-up interview was conducted with the four students as a posttest. Six problem tasks were presented with the computer available to assist in sentence writing and solution. An additional six problems with the same semantic structure as the first six were given with paper/pencil and physical manipulative objects to help with sentence writing and solution. The order of presentation was balanced. Two children received the computer tasks first and paper/pencil tasks second and the two other children received the tasks in reverse order. Two of the children used the computer with Set 1, and two used the computer with Set 2. The 12 verbal problems are listed in Table 4.

Table 3

Timing of Individual Lessons

	Lesson								
Subjects	1	2	3	4	5	6	7	8	9
Jack	4/29	5/11	5/12	5/16	5/17	5/19	5/20	5/21	5/25
Roger	4/29	5/3	5/11	5/14	5/17	5/18	5/20	5/21	5/24
Helen	4/28	5/3	5/12	5/16	5/17	5/19	5/20	5/21	5/25
Kathy	4/28	5/4	5/13	/514	5/17	5/18	5/20	5/21	5/25

Table 4

Sentence Writing Posttest Tasks

Task Set 1	Task Set 2
1. Separate James had 15 peanuts. He fed 7 of them to a monkey. How many peanuts did James have left?	1. Compare Ellen has 7 halloween candies. Her friend Greg has 12 halloween candies, How many more candies does Greg have than Ellen?
2. Compare Amy won 8 prizes at the fair. Her brother Todd won 13 prizes at the fair. How many more prizes did Todd win than Amy?	2. Join, missing addend Robert has 8 pet fish in his tank. How many more fish does he have to put in the tank so there will be 14 fish altogether?
3. Join Fred had 4 flowers. Then he picked 7 more flowers. How many flowers did Fred have altogether?	3. Separate Jeanne had 13 buttons. She gave 9 buttons to Evelyn. How many buttons did Jeanne have left?
4. Separate, missing minuend Charles had some marbles. He lost 5 of them while playing a game. Then he had 7 marbles left. How many marbles did Charles have before the game?	4. Join Norman had 6 books. His friend gave him 9 more books. How many books did Norman have altogether?
5. Join, missing addend Tony has 8 toy cars. How many more toy cars does he have to buy to have 12 cars altogether?	5. Combine, missing part There are 19 children in the class. Twelve of them are girls and the rest are boys. How many boys are in the class.
6. Combine, missing part There are 16 dogs in the park. Eleven of them are big and the rest are little. How many little dogs are in the park?	6. Separate, missing minuend There were some birds sitting on a wire. Four of the birds flew away. Then there were 7 birds left. How many birds were sitting on the wire before any flew away?

Note. Tasks are listed in the order presented to subjects.

Individual Student Results

This section contains brief anecdotal reports for each of the four subjects. Collective results on the posttest follow those reports. Overall conclusions are presented in the following section.

Jack

At the beginning of the school year, Jack had been placed in the "better" of the two mathematics classes of first-grade pupils. On the screening tasks, he used a Counting On from Larger strategy. However, when given less familiar problems, Jack resorted to use of direct modeling strategies that call for use of physical objects. His choice of strategies demonstrated that he understood the structure of all problems except the standard Compare problem.

The initial lesson presented little or no difficulty to Jack, although, as might be expected with children of his age, he demonstrated complete unfamiliarity with the computer keyboard. He did not make use of the patterned TILE configurations of the squares shown on the video display to quickly ascertain the numerosity of the display. In fact, he tended to visually scan the display vertically rather than in a horizontal fashion as might be suggested by the arrangement of the squares. Because of the relatively long period of time between the first and second lesson, Jack required extensive review of the computer procedures in the second lesson. However, the larger numbers used in the problems had no deleterious effect on his ability to solve the problems posed. When shown how to enter the displays using numeral keys rather than the $\boxed{\rightarrow}$ key, Jack's expressive face registered a large measure of both interest and delight. Midway through the teaching sequence, Jack seemed to be feeling more at ease and was catching on to the spatial configuration of the displayed squares and was using quick procedures for counting by 5s and 10s, as well as more counting on.

When complete sentence writing was taught, Jack mastered the technique despite some slight difficulty with it at first and occasional errors during the lessons. Actually, the sporadic difficulties evidenced were due more to a misunderstanding of the problem structure than to lack of the skill of entering the symbols in the correct sequence on the machine. Jack was like the other three subjects in that he had absolutely no trouble understanding the need for the SHIFT key for certain entries. Physically, he carried out the execution of the SHIFT key and the next key depression by using two different fingers of the same hand rather than one hand for SHIFT and the other hand for the desired key.

Jack accepted very readily the use of the $\square$ to represent the unknown number of the problem in the open sentence being written. This same notation was being taught during regular classroom instruction for canonical sentences ($a + b = \square$; $a - b = \square$). When presented with verbal problems other than the canonical Join and Separate ones, Jack wrote appropriate open sentences to model those problems that reflected the semantic structure of the problems and not a transformed canonical sentence. For example, for a Join, missing addend problem, Jack entered a sentence such as $3 + \square = 14$. The computer program generated for this teaching experiment does not give a visual display that makes it easy to solve such a sentence. However, Jack was able to solve most sentences of this type using methods such as counting on with the aid of fingers. In summary, Jack appeared to have learned without great difficulty the particular skills embodied in the teaching experiment. The mechanical features of keyboard entry as well as certain limitations of the computer program in terms of number size and noncanonical subtraction open-sentence gave him no problem.

On the posttest Jack performed very well. On the tasks to be performed without the computer, he was able to use paper and pencil to write correct sentences for all six tasks. For the Compare problem he wrote a separating sentence ($13 - 8 = \square$) and used a separation solution strategy. For the other five problems, sentences reflected the semantic structure. His performance on the noncanonical problems and sentences, for which little or no formal instruction had been given, was especially interesting. For the Separate, missing minuend problem, Jack wrote $\square - 5 = 7$. After thinking awhile, and referring back to the sentence he had written, Jack finally employed a trial and error strategy to solve the problem. This strategy is consistent with the semantic structure of the problem and the number sentence. Jack took much more time to solve these problems after instruction than he had on the screening tasks. He tended to resort to complete modeling more at this time, even on problems he had solved with a more advanced counting strategy prior to instruction.

On the six tasks for which he was allowed to use the computer, Jack also did well. Again, on the Separate, missing minuend problem, he tried to enter in the computer a sentence of the same form as shown above. Since the computer program would not accept such a sentence, Jack was thwarted from writing a complete open sentence. Yet, based on the partial sentence he had written, Jack again used a trial and error strategy, using counting skills and fingers.

Roger

In general, Roger exhibited highly agitated behavior, rarely sitting still. He had difficulty attending to a task, often needing to be called back to attention.. On the other hand, Ray at times showed examples of extremely keen insight into problems and their solutions. He was very friendly and outgoing, obviously enjoying his perceived good fortune at being selected for participation in the experiment.

On the screening tasks, Roger demonstrated his ability to comprehend the semantic structure of the various problems presented by choosing appropriate strategies for solution. He tended to select direct modeling strategies that mirrored the structure. However, he suffered from careless behavior and often miscounted the model sets he had constructed. He showed the ability to count forward from a number other than "one" but made one careless error.

Essentially, Roger did not have any difficulty with the mechanics of using the computer. Because of his tendency to let his attention wander, a greater amount of repetition was required for him than for the other subjects. However, his choice of modeling behaviors with the computer showed that he had no difficulty with the semantic structure of the problems presented. During the lessons when the numerical and operational symbols were introduced, Ray evidenced an upsurge in interest and ability to pay attention. His tendency to miscount, shown in the screening tasks and in earlier lessons, disappeared during this brief period of time. The problem types were the easier ones, perhaps accounting for this improvement in performance. In subsequent lessons, Roger's behavior reverted to periods of outstanding insights to problems mixed with other periods of inattention. When presented with nonroutine verbal problems of a noncanonical structure, Roger took more time in solving these problems, very often subjecting them to a semantic analysis based very clearly upon the instruction he was receiving in class.

The use of the SHIFT key to enter +, =, and the $\square$ was a relatively easy matter for Roger. Curiously, however, he did not seem to realize that the numerical keys in the topmost row of the keyboard are in numerical order. Rather, he often engaged in what appeared to be a random search.

On the posttest, Roger gave clear evidence of accepting the idea of writing a number sentence to represent a problem before attempting to solve it. When verbal problems were posed to be solved without the computer, Roger always first wrote a sentence and then tried to solve it using the cubes provided. As in previous instances, Roger made several mistakes in counting both with the computer and without it. By and large, the sentences he wrote were canonical ones with the exception of the Join, missing addend problem. On the computer he entered the sentence $8 + \square = 14$. To solve this sentence, he counted on from 8 to 14. As his device for keeping track of the number of counting words he uttered, he used the keys Q W E R T Y on the keyboard, finally counting those keys to get his correct answer of 6.

For the missing addend problem without the computer, he first wrote 8 + = $\boxed{12}$, using the space rather than the box to represent the unknown. He solved the problem using an adding on strategy. Then he said "Oh, that's supposed to be take away." and changed the + to a −, and wrote his answer in the space he had left. In other words, he initially wrote a correct open sentence and solved the problem using a process that was consistent with the sentence and the problem structure. In his regular mathematics class, Roger had been taught to analyze word problems in terms of part-whole relationships and to write canonical subtraction sentences when one of the parts was missing. It appears that Roger recognized that the answer was one of the parts and concluded that a subtraction sentence was called for.

The one incorrect sentence Roger wrote was for the missing minuend problem (Problem 4). He wrote "7 − 5 = $\square$." Subsequently he attended to his number sentence rather than the problem, constructing a set of 7 and removing 5. Thus, Ray appeared to attend to the number sentences and regarded them as something that he used in solving a word problem.

Helen

Helen gave very clear explanations of her strategies. On the screening tasks, she was generally successful, missing only the noncanonical comparison problem number 3 (see Table 1). On almost all problems, she used an advanced counting strategy of counting on, with her fingers serving as the tracking mechanism. However, on the two Separate problems, she used a simpler direct modeling strategy using the cubes provided to implement the Separating From strategy. On the three sentence-writing tasks, she managed to write a correct sentence only for the canonical subtraction problem. She did, however, determine the correct solution for all three problems.

Helen did extremely well during the first lesson. She caught on quickly to the functions of the various keys that were introduced to her. Her solution by counting the displayed objects on the video monitor indicated she was using the same counting on techniques she used in the screening tasks. In the second lesson, Helen did not do as well, perhaps due to the fact that larger numbers were involved in many of the problem situations given to her. She was much more methodical in her solution methods, and tended to make more counting errors. During the several lessons when the numerical and operational symbols were introduced, Helen performed at a very satisfactory level. She learned how to enter and use the symbols correctly to get number sentences, although almost all of them related to simple canonical situations. When the $\square$ was introduced to represent the missing number, Helen used noncanonical sentences to represent different types of problem situations. During instruction, her choice of sentence and solution method showed that Helen was strongly influenced by the semantic structure of the verbal problems being solved. Through all the lessons, Helen clearly showed that the use of the computer keyboard and the accompanying video display presented no real problems to her, either mechanically or conceptually.

On the posttest, Helen was the one subject who did not consistently use number sentences to solve problems. For three of the problems without the computer, Helen solved the problem before she wrote a number sentence. For the compare problem with the computer, she entered the incorrect sentence "8 + 13 = 5," using the sets of 8 and 13 generated by the computer for the addition sentence to solve the problem by matching.

In general Helen was influenced by the structure of the problem and modeled the action or relationships described in the problem. On three of the six problems presented without the computer, she used tally marks, a strategy she had not used on the screening tasks or during instruction.

Kathy

Kathy turned out to be a very apt pupil. She displayed the ability to pay attention to the task at hand for sustained periods of time, even when the problem task proved initially difficult for her. Of the four subjects, Kathy was the quickest to recognize the special spatial configurations of the squares on the video display. She used the configurations to her advantage when counting the displays, rarely taking the time to count all the objects shown. She would mentally move squares back and forth to make completed groupings of fives and tens which she would then count as a complete totality. For example, on a problem involving the counting of 12 squares in sector *a* and 19 squares in sector *b*, Kathy quickly gave the correct total of 31. Upon questioning how she determined the answer so rapidly, Kathy told of mentally moving one of the two lower squares in the 12-configuration over to the right side to make the 19 into a 20, whereupon she could count 10, 20, 30, and the one lone square left from the 12-configuration to make the total of 31. This ability was combined with a related problem-solving technique of using derived number facts. For example, on the screening tasks, she responded that 13 - 9 was "four" because 13 - 10 is 3 and since 9 is one less than 10, the answer must be one more, which is 4. On all the other screening tasks, Kathy used direct modeling problem solving procedures, although she was capable of more sophisticated counting procedures.

The content and skills in the initial lessons were rather easy for Kathy. She quickly learned the function and use of the specific keys and then applied them to the solution of the verbal problems posed. Almost at once, she recognized the patterned configuration of the displayed squares, as she easily determined the numerosity of particular sets of squares. The larger numbers embodied in the problems of the second lesson did not faze her. This is not to imply that Kathy solved every problem immediately and correctly. From time to time, she miscounted sets of objects or incorrectly interpreted a verbal problem.

The lessons in which the numerical and operational symbols were introduced and practiced also went well. Kathy had little or no difficulty with the mechanical and conceptual aspects of this portion of the experiment. As before, she used the special configurations to her advantage. Writing open sentences with the $\square$ was a skill that came easily to Kathy, and she exhibited the natural tendency shown by the others to write literal translations that reflected the semantic structure of the problems, rather than always writing canonical sentences that are more easily solved. In later lessons when Kathy received more experience in solving verbal problems, she became better at analyzing and solving them, and when the number sizes were small enough to be in the "basic fact" domain, she very often opted to sol them quickly by means of direct fact recall or derived fact strategies rather than take the time to write a sentence with the computer.

On the posttest Kathy clearly listened to the problems posed, analyzed them, and then wrote appropriate open sentences for them, whether the computer was available or not. When the computer was not available, Kathy used cubes for all problems, and generally modeled the number sentences she wrote. For the Compare problem, she wrote $12 - 7 = \square$ and used a separating strategy.

Posttest Summary

Sentence writing performance for the posttest is summarized in Table 5. Three of the four experimental subjects consistently wrote appropriate number sentences and solved the problems using strategies that were consistent with their number sentences. Although the fourth subject did not write number sentences for three of the problems solved without the computer, only once did she write a number sentence that was inconsistent with her solution strategy.

This performance is in marked contrast to that on the screening tasks (Table 2). Before the experimental unit, none of the four subjects wrote correct sentences for the Compare or Join, missing addend problems (screening tasks 8 and 9). They consistently ignored their incorrect sentences and directly modeled the action or relationship in the

Table 5

Sentence-Writing Performance on Posttest

Problem Type	Number of Correct Sentences Written Before Solving (Maximum = 4)	
	With Computer	Without Computer
Change/Join	4	4
Change/Separate	4	4
Comparison	3	3
Combine, missing part	4	4
Change/Join, missing addend	4	3[a]
Change/Separate, missing minuend	3	3

Note. Wording of tasks is given in Table 4.

[a]One child initially wrote a correct sentence but changed it after solving the problem.

problem. On the posttest, three of the four subjects wrote correct number sentences for the compare problems and solved the problems using a strategy that modeled their number sentence rather than the structure of the problem. The same three children also wrote correct noncanonical open sentences for the missing addend problem, although one of them subsequently altered his correct sentence. They were even generally successful in writing correct sentences for the missing minuend problem, on which almost no instruction was given.

General Conclusions

The major objective of the instructional program tested in this pilot study was to teach first-grade children to represent and solve a variety of addition and subtraction word problems. Instruction was designed to help children understand the connection between the informal strategies that they naturally invent to solve word problems and the number sentences that they are taught to write to represent them. The results of this initial pilot study strongly support the conclusion that an instructional program based on principles underlying the pilot study would be effective in teaching representational and formal problem-solving skills for solving addition and subtraction word problems.

Prior to instruction, all of the four experimental subjects wrote inappropriate number sentences for all but the most straightforward addition and subtraction problems. Furthermore, they generally viewed the number sentences as unrelated to their solution processes and ignored the sentences they wrote when solving the problem, arriving at a solution by directly modeling the action or relationships described in the problem.

Following instruction all four subjects could write number sentences to represent most problem situations and successfully used this ability to solve a variety of problems using the computer. Three of the four subjects transferred this ability to problems without the computer. To solve a simple word problem, they would first write a number sentence and then use a solution process that modeled the number sentence not the structure of the problem.

One of the factors that significantly facilitated children's ability to represent and solve certain word problems was instruction on writing noncanonical open sentences (e.g., $5 + \square = 13$ and $\square + 5 = 13$). These sentences allow children to write number sentences that are consistent with the semantic structure of missing addend problems. It has been clearly documented that young children solve missing addend problems using an adding on or counting up process which is most closely represented by an open sentence of the form $a + \square = b$ (Carpenter & Moser, 1982; in press). Blume (1981) has demonstrated that children solve these open sentences using the same adding on and counting up procedures that they use to solve missing addend word problems. This body of research strongly suggests that initial instruction on addition and subtraction should include noncanonical sentences. The results of the pilot study strongly support this conclusion.

During and following instruction, all four subjects consistently wrote noncanonical sentences to represent missing addend word problems.

In fact, they also wrote sentences like $\square - 4 = 7$ to represent missing minuend problems (see Table 4), even though they received very little instruction in this type of number sentence.

The improvement in representational skills, however, was not totally a function of learning about noncanonical number sentences. Following instruction, the children were also generally more consistent in writing appropriate canonical sentences for a variety of problems and using these sentences as a basis for solving problems. This improvement is most conspicuous for the compare problems. On the screening tasks all four children wrote an incorrect number sentence which they promptly ignored in solving the problem. On the posttest, three of the four wrote a correct canonical subtraction sentence which served as a basis for solving the problem.

Students' performance during the lessons also supports the conclusion that the instruction was successful in developing representational skills and helping the children understand the relationship between their informal strategies and the formal mathematical representations. Children quickly grasped the concepts presented to them and were almost immediately able to use them to solve problems. Although some problems were occasionally difficult for them, children were almost never totally confused or ready to give up. They genuinely seemed to understand what they were doing and believed that this insight gave them the power to correct their own errors and to solve problems that were not familiar to them.

With regard to the mechanical aspects of the children's interaction with the computer, the results can be characterized unambiguously as positive. All four first graders demonstrated their ability to work with an unfamiliar machine without any difficulty. No mechanical or motor coordination problems were detected. They understood the limitations the program imposed upon their decisions and actions (e.g., no number larger than 30, no noncanonical subtraction sentences) and worked accordingly. They experienced no difficulty using features like the shift key for upper case symbols.

The pilot study pointed out certain revisions of the computer software that are needed. Currently the program does not provide a direct solution for missing addend sentences ($a + \square = b$ and $\square + a = b$). To solve these problems, students were required to use fingers or some external counting procedure. During instruction we generally asked students to validate their answers by writing the appropriate addition sentence involving their answer. We plan to revise the program so that a visual display is produced that can be used to solve the problem. For example, consider the sentence $5 + \square = 13$. Five boxes will appear in sector a after the student has entered 5 and +. When the student has entered the complete sentence, 8 blocks will appear in sector b and the arrow will point to the empty box in that sector for the student to fill in the answer. Thus, writing the open sentence will generate the add on procedure, making an initial set of 5 and subsequently adding blocks until there is a total of 13.

Another revision calls for the inclusion of noncanonical subtraction sentences. A decision was made early in the development of the program to permit no noncanonical sentences with the operation of subtraction ($a - \square = c$ and $\square - b = c$), under the assumption that problems for which these sentences would be appropriate would be too difficult for first-grade children to understand. This assumption proved to be incorrect. Thus, a revision in future programs will be to allow all possible sentences to be written.

Another inconsistency of the program was that the displays of boxes were automatically generated if a two-digit number was entered whereas they were not so generated for a one-digit number. A conscious decision by the child had to be made to generate a display for a one-digit number by means of depressing another key (RETURN, +, -, or =). The program will be revised so that no display will appear for either one- or two-digit numbers until an additional operation has been performed.

In conclusion, further investigation seems warranted. The computer appears to allow children to rely upon their informal mathematics in an area of formal mathematics such as sentence writing. As we have argued before, the use of verbal problems does seem natural to young children because they are able to solve them in their own informal ways. The present experiment demonstrates that the computer may allow them to represent those problems in a formal way, even though they have not yet completely learned the formal algorithms and number facts. These findings suggest that instruction could be changed to make better use of children's natural ability to solve verbal problems in learning the formal mathematics of addition and subtraction. This pilot investigation suggests that the microcomputer can have an important role in that instruction.

References

Blume, G. L. An analysis of children's problem-solving behavior on verbal addition and subtraction problems. Unpublished doctoral dissertation, University of Wisconsin, Madison, 1981.

Brush, L. R. Preschool children's knowledge of addition and subtraction. Journal for Research in Mathematics Education, 1978, 9, 44-54.

Carpenter, T. P. Initial instruction in addition and subtraction: A target of opportunity for curriculum development. Proceedings of the NSF directors conferences, Washington, D.C., 1981.

Carpenter, T. P., Blume, G. L., Hiebert, J., Anick, C. M., & Pimm, D. J. Analysis and synthesis of research on children's number concepts and skills (Theoretical Paper). Madison: Wisconsin Center for Education Research, in press.

Carpenter, T. P., Hiebert, J., & Moser, J. M. Problem structure and first-grade children's initial solution processes for simple addition and subtraction problems. Journal for Research in Mathematics Education, 1981, 12, 27-39.

Carpenter, T. P., & Moser, J. M. The development of addition and subtraction problem solving skills. In T. P. Carpenter, J. M. Moser, & T. A. Romberg (Eds.), Addition and subtraction: A cognitive perspective. Hillsdale, NJ: Lawrence Erlbaum Associates, 1982.

Carpenter, T. P., & Moser, J. M.. The acquisition of addition and subtraction concepts. In R. Lesh & M. Landau (Eds.), Acquisition of mathematics concepts and processes. New York: Academic Press, in press.

Carpenter, T. P., Moser, J. M., & Hiebert, J. The effect of instruction on first-grade children's solutions of basic addition and subtraction problems (Working Paper No. 304). Madison: Wisconsin Research and Development Center for Individualized Schooling, 1981.

Carpenter, T. P., Moser, J. M., & Romberg, T. A. (Eds.). Addition and subtraction: A cognitive perspective. Hillsdale, NJ: Lawrence Erlbaum Associates, 1982.

Gibb, E. G. Children's thinking in the process of subtraction. Journal of Experimental Education, 1956, 25, 71-80.

Glaser, R., & Pellegrino, J. W. Uniting cognitive process theory and differential psychology: Back home from the wars. Intelligence, 1978, 2, 305-319.

Greeno, J. G. Some examples of cognitive task analysis with instructional implications. Paper presented at the ONR/NPRDC Conference, San Diego, California, March 1978.

Groen, G., & Parkman, J. M. A chronometric analysis of simple addition. Psychological Review, 1972, 79, 329-343.

Groen, G., & Resnick, L. B. Can preschool children invent addition algorithms? Journal of Educational Psychology, 1977, 69, 645-652.

Grouws, D. A. Differential performance of third-grade children in solving open sentences of four types (Doctoral dissertation, University of

Wisconsin, Madison, 1971). Dissertation Abstracts International, 1972, 32, 3860A. (University Microfilms No. 72-1028).

Hatano, G. Learning to add and subtract: A Japanese perspective. In T. P. Carpenter, J. M. Moser, & T. A. Romberg (Eds.), Addition and subtraction: A cognitive perspective. Hillsdale, NJ: Lawrence Erlbaum Associates, 1982.

Hiebert, J. Young children's solution processes for verbal addition and subtraction problems: The effect of the position of the unknown set. Paper presented at the annual meeting of the National Council of Teachers of Mathematics, St. Louis, 1981.

Jerman, M. Problem length as a structural variable in verbal arithmetic problems. Educational Studies in Mathematics, 1973, 5, 109-123.

Lindvall, C. M., & Ibarra, C. G. An analysis of incorrect procedures used by primary grade pupils in solving open addition and subtraction sentences. Journal for Research in Mathematics Education, 1980, 11, 50-62.

Riley, M. S., Greeno, J. G., & Heller, J. I. Development of children's problem-solving ability in arithmetic. In H. P. Ginsburg (Ed.), The development of mathematical thinking. New York: Academic press, in press.

Rosenthal, D. J., & Resnick, L. B. Children's solution processes in arithmetic word problems. Journal of Educational Psychology, 1974, 66, 817-825.

Suppes, P., & Groen, G. Some counting models for first-grade performance data on simple facts. In J. M. Scandura (Ed.), Research in mathematics education. Washington, D.C.: National Council of Teachers of Mathematics, 1967.

Suppes, P., Loftus, E. F., & Jerman, M. Problem solving on a computer-based teletype. Educational Studies in Mathematics, 1969, 2, 1-15.

Vergnaud, G. A classification of cognitive tasks and operations of thought involved in addition and subtraction problems. In T. P. Carpenter, J. M. Moser, & T. A. Romberg (Eds.), Addition and subtraction: A cognitive perspective. Hillsdale, NJ: Lawrence Erlbaum Associates, 1982.

Woods, S. S., Resnick, L. B., & Groen, C. J. An experimental test of five process models for subtraction. Journal of Educational Psychology, 1975, 67, 17-21.

Appendix A

A LISTING OF THE COMPUTER PROGRAM

```
(*$S+*)
(*$V-*)
PROGRAM MATHBOXES;

USES ANIMATION,APPLESTUFF,CHRISTUFF,TURTLEGRAPHICS;

CONST  MAXBOX = 30;

TYPE   KEYTYPE = (CLR,DUPS,NUMERAL,ESC,LEFTA,RIGHTA,CR,SPACE,MINUS,PLUS,EQUALS);
       SCREENSIDE = (RT,MID,LT,ANS);
       ARROWPOS = (LEFTBOX,MIDDLE,RIGHTBOX,ANSWER);
       ARROWDIR = (CURSORup,CURSORdown,CURSORright,CURSORleft);

VAR    RTBOX : PACKED ARRAY[0..MAXBOX,0..2] OF 0..279;
       LTBOX : PACKED ARRAY[0..MAXBOX,0..2] OF 0..279;
       NONUMERAL,OPENBOX,ADDSENTENCE,SUBSENTENCE,HELPFREEZE,OVER:   BOOLEAN;
       X,Y,I,NUMBER:   INTEGER;
       STR: STRING;
       OPENWHERE,WHERE: SCREENSIDE;
       BOXES: TEXT;
       FRAME0,FRAME1,FRAME2,FRAME3,FRAME4,FRAME5,FRAME6,FRAME7,
       FRAME8,FRAME9,FRAME0,FRAMEplus,FRAMEminus,FRAMEblank,
       FRAMEequal,FRAMEup,FRAMEdown,FRAMEright,FRAMEleft:   picture;

PROCEDURE ERRORMSG;
(**************************************************************)
(*                                                            *)
(*   ERRORMSG - errors come here.                             *)
(*                                                            *)
(**************************************************************)
BEGIN
  RINGBELL;
END;

PROCEDURE CURSOR(DIRECTION: ARROWDIR; POSITION: ARROWPOS);
(**************************************************************)
(*                                                            *)
(*   CURSOR - position arrow                                  *)
(*                                                            *)
(**************************************************************)
var X,Y,TIME: INTEGER;
BEGIN
 TIME:=0; Y:=17;

  VIEWPORT(0,279,40,56);
  FILLSCREEN(BLACK);
  VIEWPORT(0,279,0,191);

 CASE POSITION OF
   LEFTBOX: BEGIN X:=4; END;
   MIDDLE : BEGIN X:=1 ; END;
   RIGHTBOX:BEGIN X:=20; END;
   ANSWER : BEGIN X:=76; END;
 END; (* case *)

 CASE DIRECTION OF
   CURSORup:    ANIMATE(FRAMEup,X,Y,TIME);
   CURSORdown:  ANIMATE(FRAMEdown,X,Y,TIME);
   CURSORright: ANIMATE(FRAMEright,X,Y,TIME);
   CURSORleft:  ANIMATE(FRAMEleft,X,Y,TIME);
 END; (* case *)

 GOTOXY(40,23);
```

```
PROCEDURE CLEARNUM(WHERE: SCREENSIDE);
(**************************************************************)
(*                                                          *)
(*  CLEARNUM - clear numeral area.                          *)
(*                                                          *)
(**************************************************************)
VAR TIME,X1,X2,Y: INTEGER;
BEGIN
TIME:=0; Y:=20;
  CASE WHERE OF
    LT: BEGIN  X1:=3; X2:=5; END;
    RT: BEGIN  X1:=19; X2:=21; END;
    ANS: BEGIN X1:=34; X2:=36; END;
  END; (* case *)
  ANIMATE(FRAMEblank,X1,Y,TIME);
  ANIMATE(FRAMEblank,X2,Y,TIME);
END;


PROCEDURE PICTURES;
(**************************************************************)
(*                                                          *)
(*  PICTURES - set up graphic numbers.                      *)
(*                                                          *)
(**************************************************************)
BEGIN
  BLOCK(FRAMEbox,'-""=,:..*',Stav);
  BLOCK(FRAME1,'AB/GH',Stav);
  BLOCK(FRAME2,'CD/IJ',Stav);
  BLOCK(FRAME3,'EF/KL',Stav);
  BLOCK(FRAME4,'MN/ST',Stav);
  BLOCK(FRAME5,'OP/UV',Stav);
  BLOCK(FRAME6,'QR/WX',Stav);
  BLOCK(FRAME7,'YZ/ef',Stav);
  BLOCK(FRAME8,'ab/gh',Stav);
  BLOCK(FRAME9,'cd/ij',Stav);
  BLOCK(FRAME0,'kl/qr',Stav);
  BLOCK(FRAMEplus,'mn/st',Stav);
  BLOCK(FRAMEminus,'op/uv',Stav);
  BLOCK(FRAMEblank,'%&/()',Stav);
  BLOCK(FRAMEequal,'wx/01',Stav);
  BLOCK(FRAMEup,'../23',Stav);
  BLOCK(FRAMEright,'45/89',Stav);
  BLOCK(FRAMEleft,'67/  ',Stav);
  BLOCK(FRAMEdown,'""/#$',Stav);
END;


PROCEDURE DRAWNUMBER(NUMBER: INTEGER; WHERE: SCREENSIDE);
(**************************************************************)
(*                                                          *)
(*  DRAWNUMBER - place number on screenside.                *)
(*                                                          *)
(**************************************************************)
VAR TENS, UNITS : INTEGER;
    TEN:  BOOLEAN;

   PROCEDURE DRAWDIGIT(TEN:BOOLEAN; NUMBER: INTEGER; WHERE: SCREENSIDE);
   VAR TIME,X,Y: INTEGER;
   BEGIN
     TIME:=0; Y:=20;
     CASE WHERE OF
```

```
50 [illegible]
    END; (* case *)

    CASE NUMBER OF
      0: ANIMATE(FRAME0,X,Y,TIME);
      1: ANIMATE(FRAME1,X,Y,TIME);
      2: ANIMATE(FRAME2,X,Y,TIME);
      3: ANIMATE(FRAME3,X,Y,TIME);
      4: ANIMATE(FRAME4,X,Y,TIME);
      5: ANIMATE(FRAME5,X,Y,TIME);
      6: ANIMATE(FRAME6,X,Y,TIME);
      7: ANIMATE(FRAME7,X,Y,TIME);
      8: ANIMATE(FRAME8,X,Y,TIME);
      9: ANIMATE(FRAME9,X,Y,TIME);
    END; (* case *)
  END; (* drawdigit *)

BEGIN
  IF NOT NUMERAL THEN EXIT(DRAWNUMBER);
  CLEARNUM(WHERE);
  TENS:=NUMBER DIV 10;
  UNITS:=NUMBER-TENS*10;
  IF TENS <> 0 THEN BEGIN TEN:=TRUE; DRAWDIGIT(TEN,TENS,WHERE); END;
  TEN:=FALSE; DRAWDIGIT(TEN,UNITS,WHERE);
END;

PROCEDURE DRAWBOX(X,Y: INTEGER; COLOR: SCREENCOLOR);
(******************************************************************)
(*                                                                *)
(*  DRAWBOX -place box at screen location.                        *)
(*                                                                *)
(******************************************************************)
BEGIN
  VIEWPORT(X,X+7,Y,Y+7);
  FILLSCREEN(COLOR);
  VIEWPORT(0,279,0,191);
END;


PROCEDURE CREATEARRAY(WHERE: SCREENSIDE);
(**********************************************************************)
(*                                                                    *)
(* CREATEARRAY - create rtbox and ltbox arrays.                       *)
(*                                                                    *)
(**********************************************************************)
VAR X,Y,BOXNUM : INTEGER;
BEGIN

  CASE WHERE OF
  LT:   BEGIN
          X:=2; Y:=167;
          FOR BOXNUM:=1 TO MAXBOX DO
          BEGIN
            LTBOX[BOXNUM,0]:=0;
            LTBOX[BOXNUM,1]:=X;
            LTBOX[BOXNUM,2]:=Y;
            IF X > 43
              THEN
               BEGIN
                  X:=2 ;
                  IF (((BOXNUM) MOD 10)=0)   THEN Y:=Y-24
                                             ELSE Y:=Y-16;
               END
```

```
      END;
      LTBOX[0,0]:=1;
     END;

 RT:  BEGIN
       X:=115; Y:=167;
       FOR BOXNUM:=1 TO MAXBOX DO
       BEGIN
         RTBOX[BOXNUM,0]:=0;
         RTBOX[BOXNUM,1]:=X;
         RTBOX[BOXNUM,2]:=Y;
         IF X > 157
          THEN
           BEGIN
              X:=115 ;
              IF (((BOXNUM) MOD 10)=0)  THEN Y:=Y-24
                                        ELSE Y:=Y-16;
           END
          ELSE X:=X+14;
         DRAWBOX(RTBOX[BOXNUM,1],RTBOX[BOXNUM,2],BLACK);
       END;
       RTBOX[0,0]:= 1;
      END;

   END; (* case *)

END;

FUNCTION  GETKEY(OKSET: SETOFCHAR): CHAR;
(*****************************************************************)
(*                                                               *)
(* GETKEY - get a character from keyboard.                       *)
(*                                                               *)
(*****************************************************************)

VAR CH:CHAR; GOOD:BOOLEAN;

BEGIN
 REPEAT
  READ(KEYBOARD,CH);
  IF EOLN(KEYBOARD) THEN CH:=CHR(13);
  IF CH='Q' THEN BEGIN STDFONT; EXIT(PROGRAM); END;
  GOOD:=CH IN OKSET;
  IF NOT GOOD THEN RINGBELL
 UNTIL GOOD;
 GETKEY:=CH
END;

PROCEDURE KEYCHECK(VAR KEY: CHAR; VAR CLASS: KEYTYPE);
(*****************************************************************)
(*                                                               *)
(* KEYCHECK - Determine what kind of key was pressed.            *)
(*                                                               *)
(*****************************************************************)

VAR SPSET,DIGITS: SET OF CHAR;

BEGIN
GOTOXY(41,27);
DIGITS:=['0'..'9'];
(*      ret      esc       ->       <-       sp
         -        +        =        ?        X         x     *)
SPSET:=[CHR(13),CHR(27),CHR(21),CHR(8),CHR(32),
        CHR(45),CHR(43),CHR(61),CHR(63),CHR(88),CHR(120)];
```

```
  IF KEY IN SPSET
  THEN
    BEGIN
      IF KEY=CHR(13) THEN CLASS:=CR;
      IF KEY=CHR(27) THEN CLASS:=ESC;
      IF KEY=CHR(8)  THEN CLASS:=LEFTA;
      IF KEY=CHR(21) THEN CLASS:=RIGHTA;
      IF KEY=CHR(32) THEN CLASS:=SPACE;
      IF KEY=CHR(45) THEN CLASS:=MINUS;
      IF KEY=CHR(43) THEN CLASS:=PLUS;
      IF KEY=CHR(61) THEN CLASS:=EQUALS;
      IF KEY=CHR(63) THEN CLASS:=QUES;
      IF ((KEY=CHR(88)) OR (KEY=CHR(120))) THEN CLASS:=CLR;
    END
  ELSE
    BEGIN
      CLASS:=NUMERAL;
    END;

END;


PROCEDURE ADDBOX(WHERE: SCREENSIDE);
(**************************************************************)
(*                                                            *)
(*  ADDBOX - place box on screen using appropriate side.      *)
(*                                                            *)
(**************************************************************)
VAR BOXNUM: INTEGER;

BEGIN
  CASE WHERE OF

  LT: BEGIN
        BOXNUM:=LTBOX[0,0];
        IF BOXNUM <> 0 THEN
          BEGIN
            DRAWBOX(LTBOX[BOXNUM,1],LTBOX[BOXNUM,2],BLUE);
            LTBOX[0,0]:=BOXNUM+1;    (* next free box *)
            IF LTBOX[0,0] > MAXBOX THEN LTBOX[0,0]:=0;   (* none left *)
            LTBOX[BOXNUM,0]:=-1;         (* filled *)
          END;
      END;

  RT: BEGIN
        BOXNUM:=RTBOX[0,0];
        IF BOXNUM <> 0 THEN
          BEGIN
            DRAWBOX(RTBOX[BOXNUM,1],RTBOX[BOXNUM,2],GREEN);
            RTBOX[0,0]:=BOXNUM+1;    (* next free box *)
            IF RTBOX[0,0] > MAXBOX THEN RTBOX[0,0]:=0;   (* none left *)
            LTBOX[BOXNUM,0]:=-1;         (* filled *)
          END;
      END;

  END; (* case *)
END;
```

61

```
PROCEDURE SUBBOX(WHERE: SCREENSIDE);
(**************************************************************)
(*                                                            *)
(*  SUBBOX - remove box from screen using the side.           *)
```

```
VAR BOXNUM: INTEGER;

BEGIN
  CASE WHERE OF

  LT: BEGIN
        BOXNUM:=LTBOX[0,0];
        IF BOXNUM <> 1 THEN
          BEGIN
            IF BOXNUM=0 THEN BOXNUM:=MAXBOX ELSE BOXNUM:=BOXNUM-1;
            DRAWBOX(LTBOX[BOXNUM,1],LTBOX[BOXNUM,2],BLACK);
            LTBOX[0,0]:=BOXNUM;   (* nxt free box *)
            LTBOX[BOXNUM,0]:=0;         (* empty  *)
          END;
      END;

  RT: BEGIN
        BOXNUM:=RTBOX[0,0];
        IF BOXNUM <> 1 THEN
          BEGIN
            IF BOXNUM=0 THEN BOXNUM:=MAXBOX ELSE BOXNUM:=BOXNUM-1;
            DRAWBOX(RTBOX[BOXNUM,1],RTBOX[BOXNUM,2],BLACK);
            RTBOX[0,0]:=BOXNUM;   (* nxt free box *)
            RTBOX[BOXNUM,0]:=0;         (* empty  *)
          END;
        END;

  END; (* case *)
END;


PROCEDURE MOVEBOX(FROM: SCREENSIDE);
(***********************************************************)
(*                                                         *)
(*  MOVEBOX - shuffle box from side to side                *)
(*                                                         *)
(***********************************************************)
VAR BOXNUM: INTEGER;

BEGIN
  CASE FROM OF

  LT: BEGIN
       IF LTBOX[0,0] <> 1 THEN
       BEGIN
         SUBBOX(LT);
         DRAWBOX(86,120,BLUE);
         DRAWBOX(86,120,GREEN);
         DRAWBOX(86,120,BLACK);
         ADDBOX(RT);
       END
       ELSE ERRORMSG;
     END;

  RT: BEGIN
       IF RTBOX[0,0]<>1 THEN
       BEGIN
         SUBBOX(RT);
         DRAWBOX(86,120,GREEN);
         DRAWBOX(86,120,BLUE);
         DRAWBOX(86,120,BLACK);
         ADDBOX(LT);
       END
       ELSE ERRORMSG;
```

```
54 ...
END;

PROCEDURE DRAWOPENBOX(WHERE: SCREENSIDE);
(*****************************************************************)
(*                                                               *)
(*  DRAWOPENBOX - place open box on screen.                      *)
(*                                                               *)
(*****************************************************************)
var X,Y: INTEGER;

BEGIN
  IF NONUMERAL THEN EXIT(DRAWOPENBOX);
  Y:=20;
  CASE WHERE OF
   LT:   X:=-3 ;
   MID:  X:=-3;
   RT:   X:=19;
  ANS:   X:=34;
  END; (* case *)

  CLEARNUM(WHERE);
  ANIMATE(FRAMEbox,X,Y,0);
  OPENWHERE:=WHERE;
END;
(*$I BOXES2.TEXT *)

(*****************************************************************)
(*                                                               *)
(*       special draw routines for operations                    *)
(*                                                               *)
(*****************************************************************)
PROCEDURE DRAWEQUALS;
BEGIN
 IF (ADDSENTENCE OR SUBSENTENCE) THEN
 BEGIN
   CURSOR(CURSORdown,ANSWER);
   WHERE:=ANS;
   IF NONUMERAL THEN EXIT(DRAWEQUALS);
   X:=28; Y:=20; ANIMATE(FRAMEequals,X,Y,0);
 END;
END;

PROCEDURE DRAWPLUS;
BEGIN
  ADDSENTENCE:=TRUE;
 IF NONUMERAL THEN EXIT(DRAWPLUS);
  X:=12; Y:=20; ANIMATE(FRAMEplus,X,Y,0);
END;

PROCEDURE DRAWMINUS;
BEGIN
  SUBSENTENCE:=TRUE;
 IF NONUMERAL THEN EXIT(DRAWMINUS);
  X:=12; Y:=20; ANIMATE(FRAMEminus,X,Y,0);
END;

PROCEDURE ESCAPE;
BEGIN
  FILLSCREEN(BLACK);
  CLEARNUM(LT);
  CREATEARRAY(LT);
  CLEARNUM(RT);
  CREATEARRAY(RT);
```

```
    ADDSENTENCE:=FALSE;
    SUBSENTENCE:=FALSE;
    OPENBOX: FALSE;
  END;

  PROCEDURE ENDNUMERAL(NUMBER: INTEGER;  WHERE: SUFFENSIDE);
  VAR I: INTEGER;
  BEGIN
    IF NUMBER - 0
    THEN DRAWNUMBER(NUMBER,WHERE) .
    ELSE FOR I:= 1 TO NUMBER DO ADDBOX(WHERE);
  END;

  PROCEDURE LEFTSIDE;
  (**************************************************************)
  (*                                                            *)
  (*          LEFTSIDE                                          *)
  (*                                                            *)
  (**************************************************************)
  (*$G+*)
  LABEL 1;

  VAR  KEY: CHAR;
       S1,STR:  STRING;
       CLASS: KEYTYPE;
       DIGITMODE,DONE:  BOOLEAN;
       NUMBER: INTEGER;

  BEGIN
    OPENBOX: FALSE; DIGITMODE:-FALSE; DONE:-FALSE;
    NUMBER:-0; STR:=''; S1:=' ';
    REPEAT
  1:  KEYCHECK(KEY,CLASS);

      IF OPENBOX
      THEN BEGIN
             IF (((CLASS=ESC) OR (CLASS-PLUS)) OR (CLASS-CLR))
             THEN  BEGIN  END
             ELSE  BEGIN ERRORMSG; GOTO 1; END;
           END;
  (*$G-*)
      CASE CLASS OF
        CR: BEGIN
              IF DIGITMODE
              THEN BEGIN ENDNUMERAL(NUMBER,WHERE); DIGITMODE:-FALSE; STR:  '1, ':EN
  D;
            END;
        CLR: BEGIN
                CREATEARRAY(LT);
                CLEARNUM(LT);
                OPENBOX:-FALSE; DIGITMODE:=FALSE; NUMBER: 0; STR: '';
             END;
        ESC:  BEGIN DONE:=TRUE; ESCAPE; END;
        LEFTA:  BEGIN
                STR:-'123';
                SUBBOX(WHERE);
                IF LTBOX[0,0] - 0
                  THEN NUMBER:=LTBOX[0,0]-1
                  ELSE NUMBER:=MAXBOX;
                IF NUMBER > 0
                  THEN DRAWNUMBER(NUMBER,WHERE)
                  ELSE BEGIN
                         CLEARNUM(WHERE);
                         STR:='';
                         DIGITMODE:=FALSE; NUMBER:=0;
```

```
    RIGHTBX: BEGIN
              STR: '123';
              ADDBOX(WHERE);
              IF LTBOX[O,O]  O
                THEN NUMBER:=LTBOX[O,O]
                ELSE NUMBER:=MAXBOX;
              IF NUMBER  O THEN DRAWNUMBER(NUMBER,WHERE);
             END;
SPACE:   BEGIN
              IF (DIGITMODE AND (STR  '123'))
              THEN BEGIN ENDNUMERAL(NUMBER,WHERE); DIGITMODE: FALSE; END;
               WHERE: RI;
               DONE:=TRUE;
             END;
MINUS:   BEGIN
                DRAWMINUS;
                IF STR  '123' THEN ENDNUMERAL(NUMBER,WHERE);
                WHERE: MID;
                DONE:=TRUE;
             END;
PLUS:    BEGIN
                DRAWPLUS;
                IF CANOT OPENBOX AND STR  THEN ENDNUMERAL NUMBER,WHER
                DONE: TRUE;
                WHERE: RI;
            END;
EQUALS:   BEGIN
                IF STR  '123' THEN ENDNUMERAL(NUMBER,WHERE);
                DRAWEQUALS;
                WHERE: ANS;
                DONE:=TRUE;
              END;
ONES:    BEGIN
              IF STR  '123' THEN
              BEGIN
                DRAWOPENBOX(WHERE);
                STR: '123';
                OPENBOX: TRUE;
              END
              ELSE ERRORMSG;
             END;
NUMERAL: BEGIN
               IF LENGTH(STR)  1
               THEN
                 BEGIN
                   SETT:=KEY;
                   STR:=CONCAT(STR,SETT); NUMBER:=VALUE(STR);
                   DRAWNUMBER(NUMBER,WHERE);
                   DIGITMODE:=TRUE;
                   IF NUMBER  MAXBOX THEN
                        BEGIN
                          NUMBER:=0;
                          STR: '';
                          ERRORMSG;
                          DIGITMODE:=FALSE;
                          CLEARNUM(LT);
                        END;
                   IF NUMBER 9 THEN
                        BEGIN
                          ENDNUMERAL(NUMBER,WHERE); DIGITMODE: FALSE;
                          STR:='123';
                        END;
                 END
               ELSE ERRORMSG;
```

```
  UNTIL DONE;
END;

PROCEDURE MIDSCREEN;
(*********************************************************)
(*                                                       *)
(*          MIDSCREEN                                    *)
(*                                                       *)
(*********************************************************)
VAR  KEY: CHAR;
     S1,STR:  STRING;
     CLASS: KEYTYPE;
     DIGITMODE,DONE:  BOOLEAN;
     RTNUMBER,LTNUMBER,NUMBER: INTEGER;


   PROCEDURE TRANSFER(NUMBER: INTEGER;  WHERE: SCREENSIDE);
   VAR I: INTEGER;
   BEGIN
     IF NUMBER  = LTNUMBER
     THEN
       BEGIN
         IF NUMBER=0 THEN DRAWNUMBER(NUMBER,RT);
         FOR I:= 1 TO NUMBER DO
         BEGIN
           MOVEBOX(WHERE);
           DRAWNUMBER(I,RT);
         END;
       END
     ELSE ERRORMSG;
   END;

BEGIN
  DIGITMODE:=FALSE; DONE:=FALSE;
  NUMBER:=0; STR:=''; S1:=' ';

  IF LTBOX[0,0]    0
    THEN LTNUMBER:=LTBOX[0,0]-1
    ELSE LTNUMBER:=MAXBOX;

  REPEAT
    KEYCHECK(KEY,CLASS);

    CASE CLASS OF
      CLR: BEGIN
              IF RTBOX[0,0]    0
                 THEN RTNUMBER:=RTBOX[0,0]-1
                 ELSE RTNUMBER:=MAXBOX:
               TRANSFER(RTNUMBER,RT);
               CREATEARRAY(RT):
               CLEARNUM(RT);
               DIGITMODE:=FALSE; NUMBER: 0; STR:='';
             END;
      CR: BEGIN
             IF DIGITMODE THEN
             BEGIN
              TRANSFER(NUMBER,LT);
              DIGITMODE:=FALSE; STR:='123';
             END;
           END;
      ESC:  BEGIN DONE:=TRUE; ESCAPE; END;
      LEFTA:  BEGIN
               STR:='123';
               MOVEBOX(RT);
```

```
            ELSE NUMBER:-MAXBOX:
          IF NUMBER -- 0 THEN DRAWNU  FR(NUMBER,RT):
         END:
RIGHT:   BEGIN
          STR:='1.3':
          MOVEBOX(LT):
          IF RTBOX[0.0] > 0
            THEN NUMBER:=RTBOX[0.0]-1
            ELSE NUMBER:=MAXBOX:
          IF NUMBER  - 0 THEN DRAWNUMBER(NUMBER.RT):
         END:
SPACE:   BEGIN
          IF DIGITMODE
          THEN BEGIN TRANSFER(NUMBER.LT): DIGITMODE:-FALSE: END:
           DRAWEQUALS:
           WHERE:-ANS:
           DONE:-TRUE:
         END:
MINUS:   BEGIN
           DRAWMINUS:
           IF (DIGITMODE OR (STR  '123'))
             THEN TRANSFER(NUMBER.LT):
           DRAWEQUALS:
           WHERE: -ANS:
           DONE:=TRUE:
         END:
PLUS.    BEGIN
           IF LENGTH(STR)-0
           THEN
            BEGIN
              DRAWPLUS:
              ENDNUMERAL(NUMBER.RT):
              DONE:-TRUE:
              WHERE:=RT:
            END
           ELSE ERRORMSG:
         END:
EQUALS:   BEGIN
           IF STR     123' THEN TRANSFER(NUMBER,LT):
           DRAWEQUALS:
           WHERE: -ANS:
           DONE:-TRUE:
          END:
QUES:    ERRORMSG:
NUMERAL: BEGIN
           IF LENGTH(STR)  = 1
           THEN
             BEGIN
               S1[1]:=KEY:
               STR:-CONCAT(STR,S1): NUMBER:-VALUE(STR):
               DRAWNUMBER(NUMBER.RT):
               DIGITMODE:-TRUE:
               IF NUMBER > LTNUMBER THEN
                 BEGIN
                   NUMBER:=0:
                   STR:='':
                   ERRORMSG:
                   DIGITMODE:=FALSE:
                   CLEARNUM(RT):
                 END:
               IF NUMBER>9 THEN
                BEGIN
                 TRANSFER(NUMBER,LT):
                 DIGITMODE:=FALSE: STR:='123':
```

```
            END
           ELSE ERRORMSG;
          END;
    END;

   UNTIL DONE;
 END;


 PROCEDURE RIGHTSIDE;
 (***************************************************************)
 (*                                                             *)
 (*         RIGHTSIDE                                           *)
 (*                                                             *)
 (***************************************************************)
 VAR  KEY: CHAR;
      S1,STR:  STRING;
      CLASS: KEYTYPE;
      DIGITMODE,DONE:  BOOLEAN;
      NUMBER: INTEGER;

 BEGIN
   DIGITMODE:=FALSE; DONE:=FALSE;
   NUMBER:=0; STR:=''; S1:=' ';
   REPEAT
     KEYCHECK(KEY,CLASS);

     CASE CLASS OF
       CLR: BEGIN
                CREATEARRAY(RT);
                CLEARNUM(RT);
                IF OPENWHERE=RT THEN OPENBOX:=FALSE;
                DIGITMODE:=FALSE; NUMBER:=0; STR:='';
               END;
       CR: BEGIN
               IF (DIGITMODE AND (STR<>'123'))
 D:            THEN BEGIN ENDNUMERAL(NUMBER,WHERE); DIGITMODE:=FALSE; STR:='123';EN
             END;
       ESC:  BEGIN DONE:=TRUE; ESCAPE; END;
       LEFTA:   BEGIN
                STR:='123';
                SUBBOX(WHERE);
                IF RTBOX[0,0] > 0
                  THEN NUMBER:=RTBOX[0,0]-1
                  ELSE NUMBER:=MAXBOX;
                IF NUMBER >= 0 THEN DRAWNUMBER(NUMBER,WHERE);
               END;
       RIGHTA:  BEGIN
                STR:='123';
                ADDBOX(WHERE);
                IF RTBOX[0,0] > 0
                  THEN NUMBER:=RTBOX[illegible]-1
                  ELSE NUMBER:=MAXBOX;
                IF NUMBER >= 0 THEN DRAWNUMBER(NUMBER,WHERE);
               END;
       SPACE:   BEGIN
                 IF STR <> '123' THEN ENDNUMERAL(NUMBER,WHERE);
                 IF (ADDSENTENCE OR SUBSENTENCE)THEN DRAWEQUALS;
                 WHERE:=ANS;
                 DONE:=TRUE;
               END;
       MINUS:   BEGIN
                 IF LENGTH(STR)=0
                 THEN
```

```
                DRAWMINUS;
                WHERE:=MID;
                DONE:=TRUE;
              END
            ELSE ERRORMSG;
          END;
    PLUS:   BEGIN
              DRAWPLUS;
              IF DIGITMODE THEN
                BEGIN
                  ENDNUMERAL(NUMBER,WHERE);
                  STR:='123';
                  DIGITMODE:=FALSE;
                END;
            END;
    EQUALS:   BEGIN
               IF STR <> '123' THEN ENDNUMERAL(NUMBER,WHERE);
               DRAWEQUALS;
               WHERE:=ANS;
               DONE:=TRUE;
              END;
    QUES    :BEGIN
             IF ((ADDSENTENCE AND (NOT OPENBOX)) AND (STR<>'123'))THEN
                   BEGIN
                    DRAWOPENBOX(WHERE);
                    STR:='123';
                    OPENBOX:=TRUE;
                    DRAWEQUALS; WHERE:=ANS; DONE:=TRUE;
                   END ;
            END;
    NUMERAL: BEGIN
               IF LENGTH(STR) <= 1
               THEN
                 BEGIN
                   S1[1]:=KEY;
                   STR:=CONCAT(STR,S1); NUMBER:=VALUE(STR);
                   DRAWNUMBER(NUMBER,WHERE);
                   DIGITMODE:=TRUE;
                   IF NUMBER > MAXBOX THEN
                       BEGIN
                         NUMBER:=0;
                         STR:='';
                         ERRORMSG;
                         DIGITMODE:=FALSE;
                         CLEARNUM(RT);
                       END;
                   IF NUMBER>9 THEN
                       BEGIN
                         ENDNUMERAL(NUMBER,WHERE); DIGITMODE:=FALSE;
                         STR:='123';
                       END;
                 END
               ELSE ERRORMSG;
             END;
  END;

 UNTIL DONE;
END;


PROCEDURE FARSIDE;
(*************************************************************)
(*                                                          *)
(*        FARSIDE                                           *)
```

```
VAR  KEY: CHAR;
     S1,STR:  STRING;
     CLASS: KEYTYPE;
     DIGITMODE,DONE:   BOOLEAN;
     LTNUMBER,RTNUMBER,SOLN,X,Y,NUMBER: INTEGER;

BEGIN
  DIGITMODE:=FALSE; DONE:=FALSE;
  NUMBER:=0; STR:=''; S1:=' ';
  REPEAT
    KEYCHECK(KEY,CLASS);

    CASE CLASS OF
      CLR: BEGIN
               CLEARNUM(ANS);
               DIGITMODE:=FALSE; NUMBER:=0; STR:=' ';
             END;
      CR: BEGIN
            IF DIGITMODE THEN
              BEGIN
                ENDNUMERAL(NUMBER,WHERE);
                DIGITMODE:=FALSE;
                STR:='123';
                IF  (ADDSENTENCE AND OPENBOX)  THEN
                   BEGIN
                     IF LTBOX[0,0] <> 0
                         THEN LTNUMBER:=LTBOX[0,0]-1
                         ELSE LTNUMBER:=MAXBOX;
                     IF RTBOX[0,0] <> 0
                         THEN RTNUMBER:=RTBOX[0,0]-1
                         ELSE RTNUMBER:=MAXBOX;
                     IF OPENWHERE=RT
                       THEN SOLN:=NUMBER-LTNUMBER
                       ELSE SOLN:=NUMBER-RTNUMBER;
                     ENDNUMERAL(SOLN,OPENWHERE);
                     DRAWNUMBER(SOLN,OPENWHERE);
                   END;
              END;
          END;
      ESC:  BEGIN DONE:=TRUE; ESCAPE; END;
      LEFTA:  BEGIN
                ERRORMSG;
              END;
      RIGHTA: BEGIN
                ERRORMSG;
              END;
      SPACE:  BEGIN
                ERRORMSG;
              END;
      MINUS:  BEGIN
                ERRORMSG;
              END;
      PLUS:  BEGIN
               ERRORMSG;
             END;
      EQUALS:  BEGIN
                 DRAWEQUALS;
               END;
      QUES:  BEGIN
               IF ((ADDSENTENCE OR SUBSENTENCE) AND (NOT OPENBOX))
               THEN  BEGIN DRAWOPENBOX(WHERE); END
               ELSE  ERRORMSG;
             END;
      NUMERAL: BEGIN
```

```
                      STR:=CONCAT(STR,S1); NUMBER:=VALUE(STR);
                      DRAWNUMBER(NUMBER,WHERE);
                      DIGITMODE:=TRUE;
                     IF NUMBER > (2*MAXBOX) THEN
                           BEGIN
                             NUMBER:=0;
                             STR:='';
                             ERRORMSG;
                             DIGITMODE:=FALSE;
                             CLEARNUM(ANS);
                           END;
                  END
                ELSE ERRORMSG;

               END;
    END;

  UNTIL DONE;
   X:=28; Y:=20; ANIMATE(FRAMEblank,X,Y,0);
   CLEARNUM(ANS);
END;


FUNCTION MENUCHOICE(PICKS:SETOFCHAR; Y:INTEGER):CHAR;
(*****************************************************************)
(*  Identifies user selection from SET OF CHARACTERS            *)
(*****************************************************************)

VAR S1: STRING; CH: CHAR; PIX: SETOFCHAR;

BEGIN
 PIX:=[];
 FOR CH:=CHR(65) TO CHR(90)
  DO IF (CH IN PICKS)
   OR (CHR(ORD(CH)+32) IN PICKS)
   THEN PIX:=PIX+[CH,CHR(ORD(CH)+32)];
 REPEAT
  GOTOXY(1,Y);
  WRITE('( ) Type letter. Then press return.');
  GOTOXY(2,Y);S1:='';READLN(S1);
  IF LENGTH(S1)=0 THEN S1:=' ';
  IF NOT (S1[1] IN PIX) THEN ERRORMSG
 UNTIL S1[1] IN PIX;
 MENUCHOICE:=S1[1];GOTOXY(40,23)
END;

PROCEDURE MENU;
(***********************************************************)
(*                                                         *)
(*       MENU SECTION                                      *)
(*                                                         *)
(***********************************************************)
VAR MAINCHAR: CHAR;
BEGIN

 FILLSCREEN(BLACK);
 GOTOXY(0,0);
 WRITELN(' Wisconsin Center for Education Research');
 WRITELN('  (c) 1982 by the Regents of');
 WRITELN('      the University of Wisconsin');
 WRITELN; WRITELN;
 WRITELN('                 MATHBOXES');WRITELN;WRITELN;
 WRITELN(' (N)umerals to be displayed.');
 WRITELN(' (W)ithout numerals.');
```

```
WRITELN('  (Q)uit.');

MAINCHAR: MENUCHOICE(['W','w','N','n','Q','q'],1,');

   CASE MAINCHAR OF
    'N','n': NONUMERALS:=FALSE;
    'W','w': NONUMERALS:=TRUE;
    'Q','q': EXIT(PROGRAM);
   END;

FILLSCREEN(BLACK);
END;

(**********************************************************************)
(*                                                                    *)
(*           M A I N     R O U T I N E                                *)
(*                                                                    *)
(**********************************************************************)
BEGIN
  MENU;
  ADDSENTENCE:=FALSE; HELLFREEZESOVER:=FALSE;
  SUBSENTENCE:=FALSE;  OPENBOX:=FALSE;
  INITTURTLE; GRAFMODE;
  LOADFONT(BOXES,'#4:BOXES.FONT');  PICTURES;  USEFONT(BOXES);
  FILLSCREEN(BLACK);
  CREATEARRAY(LT);
  CREATEARRAY(RT);

WHERE:=LT;
REPEAT

  CASE WHERE OF
   LT:  BEGIN
         CURSOR(CURSORup,LEFTBOX);
         LEFTSIDE;
        END;
   MID: BEGIN
         CURSOR(CURSORright,MIDDLE);
         MIDSCREEN;
        END;
   RT:  BEGIN
         CURSOR(CURSORup,RIGHTBOX);
         RIGHTSIDE;
        END;
   ANS: BEGIN
         CURSOR(CURSORdown,ANSWER);
         FARSIDE;
        END;

     END; (* case *)

UNTIL HELLFREEZESOVER;
STDFONT;

END.
```

ASSOCIATED FACULTY

Bradford B. Brown
Assistant Professor
Educational Psychology

Glen G. Cain
Professor
Economics

Thomas P. Carpenter
Professor
Curriculum and Instruction

Robin S. Chapman
Professor
Communicative Disorders

William H. Clune
Professor
Law

W. Patrick Dickson
Assistant Professor
Child and Family Studies

William Epstein
Professor
Psychology

Herbert A. Exum
Assistant Professor
Counseling and Guidance

Elizabeth H. Fennema
Professor
Curriculum and Instruction

Lloyd E. Frohreich
Professor
Educational Administration

Marvin J. Fruth
Professor
Educational Administration

Arthur M. Glenberg
Associate Professor
Psychology

Helen Goodluck
Assistant Professor
English and Linguistics

Maureen T. Hallinan
Professor
Sociology

J. R. Hollingsworth
Professor
History

Dale D. Johnson
Professor
Curriculum and Instruction

Carl F. Kaestle
Professor
Educational Policy Studies

Herbert J. Klausmeier
V. A. C. Henmon Professor
Educational Psychology

Joel R. Levin
Professor
Educational Psychology

Cora B. Marrett
Professor
Sociology and Afro-American Studies

Douglas W. Maynard
Assistant Professor
Sociology

Jon F. Miller
Professor
Communicative Disorders

Fred M. Newmann
Professor
Curriculum and Instruction

Michael R. Olneck
Associate Professor
Educational Policy Studies

Penelope L. Peterson
Associate Professor
Educational Psychology

Gary G. Price
Assistant Professor
Curriculum and Instruction

W. Charles Read
Professor
English and Linguistics

Thomas A. Romberg
Professor
Curriculum and Instruction

Richard A. Rossmiller
Professor
Educational Administration

Richard Ruiz
Assistant Professor
Educational Policy Studies

Peter A. Schreiber
Associate Professor
English and Linguistics

Barbara J. Shade
Associate Professor
Educational Psychology,
UW-Parkside

Marshall S. Smith
Center Director and Professor
Educational Policy Studies
and Educational Psychology

Aage B. Sorensen
Professor
Sociology

B. Robert Tabachnick
Professor
Curriculum and Instruction
and Educational Policy
Studies

Karl E. Taeuber
Professor
Sociology

Bruce A. Wallin
Assistant Professor
Political Science

Gary G. Wehlage
Professor
Curriculum and Instruction

Alex Cherry Wilkinson
Assistant Professor
Psychology

Louise Cherry Wilkinson
Associate Professor
Educational Psychology

Steven R. Yussen
Professor
Educational Psychology